REMARKS

Entry of this amendment along with reconsideration and allowance of the subject application are respectfully requested.

Claim 16 stands rejected under 35 U.S.C. §112, second paragraph. In particular, the Examiner suggests amendment to clarify that the interpreter does not execute native program instruction words. Claim 16 has been amended to remove the word "both" which should obviate the Examiner's concern. Withdrawal of the rejection under 35 U.S.C. §112, second paragraph is respectfully requested.

Claims 1-5, 8 and 19 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent 6,513,156 to Bak. This rejection is respectfully traversed.

A description of Bak's system, Bak's objectives, and distinctions of the independent claims from Bak's teachings are set forth in the prior response. Notwithstanding those distinctions, the Examiner maintains the anticipation rejection. In an effort to address concerns raised by the Examiner in paragraph 3D of the Office Action, claims 1 and 19 have been amended to specify that the return address used as a pointer to the interpreted code portion is generated *during execution* of the native code call.

In maintaining the anticipation rejection based on Bak, the Examiner relies on Figure 11 (and, in particular, step 907) and the text at column 12, lines 13-17 as allegedly teaching the features of claim integers (v), (vi), and (vii) of claims 1 and 19. This reliance is misplaced.

Bak discloses in column 11, line 42 to column 12, lines 22, the following sequence of operations.

1) A current bytecode pointer is saved.

2) An interpreter return address is pushed onto the stack memory.

3) Snippet code for the invoke_virtual function is executed. The snippet comprises a compiled native machine instruction equivalent to "call<function>", in which the address of the desired virtual function is hard-coded in the native machine instruction (see column 11, lines 50-65).
4) One the virtual function finishes execution, the system returns to the return address that was pushed onto the stack.
5) Native instructions are executed to reload the save bytecode pointer.
6) The system continues to execute non-native code from the point where the interpreter left-off.

The Examiner equates step (3) of Bak with the step of a native code portion invoking interpretation of an interpreted code portion by executing a native code call instruction to the instruction interpreter. In paragraph 8, the Examiner equates the return address of claim 1 with the return address of Bak retrieved from stack memory (corresponding to steps (2) and (4) of the above sequence). But Bak does **not** disclose that the "return address is generated *during execution* of said native code call."

The Examiner over-generalizes the language element (v) in claims 1 and 19 by stating "Bak's invention alternates between interpreted code and native code...Thus, interpreted and native code are both invoked in the executing program," (see top of page 12 of the Office Action). Notwithstanding this over-generalization, there is no question that Bak fails to teach generating the return address during execution of the native code call, "said return address specifying a location within said memory for said native code call instruction."

The independent claims are directed to the problem of providing a more efficient switch between execution of native instruction words and interpreted program instruction words. This problem is addressed by performing the switch using a native code call instruction, which during its execution generates a return address pointing to the interpreted program instruction words on which the instruction interpreter is to operate. Thus, only a single native code instruction is required to make the switch.

Bak teaches away from generating the return address *during execution* of the native function call. Rather, Bak teaches that the return address is pushed onto the stack *before* execution of the native code snippet for the invoke_virtual function and then returns to the pushed return address after execution of the virtual function whose address is hard-coded by the snippet (see Figure 11, steps 903, 905, and 907)

Thus, although Bak fails to disclose elements (v)-(vii) of the independent claims as explained in the previous response, there is no question that Bak fails to disclose or suggest element (vi) of claims 1 and 19. Accordingly, the anticipation rejection should be withdrawn. Moreover, the obviousness rejections set forth in numbered paragraphs 10 and 12 rely on secondary references which do not remedy the deficiencies of Bak with respect to the independent claims.

This application is now in condition for allowance. An early notice to that effect is earnestly solicited.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: [illegible]
John R. Lastova
Reg. No. 33,149

JRL:at
1100 North Glebe Road, 8th Floor
Arlington, VA 22201-4714
Telephone: (703) 816-4000
Facsimile: (703) 816-4100